Buză Elena-Raluca

Grupa 331AA

Tema – Tehnici de invatare automata

Recognizing handwritten digits

1. Descrierea proiectului

Aceasta aplicatie isi propune sa ilustreze procesul de recunoastere a cifrelor scrise de mana prin aplicarea unui algoritm de invatare automata, anume Naïve Bayes (NB). Exista mai multe tipuri de algoritmi Naïve Bayes, precum: Gaussian, Multinomial, Bernoulli. Pentru aceasta aplicatie am folosit Multinomial Naïve Bayes.

Recunoasterea cifrelor scrise de mana reprezinta o problema fundamentala in invatarea automata, avand aplicatii in recunoasterea optica a caracterelor si procesarea documentelor digitale.

Obiectivul principal este de a explica abordarea utilizarii algoritmului Multinomial Naïve Bayes pentru a clasifica cu precizie cifrele scrise de mana din setul de date.

2. Setul de date

Setul de date contine 10 foldere, pentru fiecare cifra (0 ... 9). Fiecare folder contine un numar variabil de imagini cu extensia .png, fiecare imagine fiind etichetata. Setul de date a fost obtinut prin descarcarea imaginilor corespunzatoare fiecarei cifre. Exista un numar mare de imagini in setul de date, tocmai pentru ca antrenarea algoritmului sa se faca intr-un mod corect.

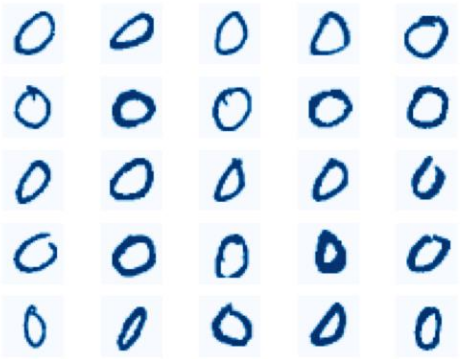
Datele sunt relevante temei alese: avem imagini cu toate cifrele. Aceastea sunt etichetate, fiecare imagine avand un nume/eticheta unic, neexistand duplicate. Sunt organizate din punct de vedere structural, caci avem 10 foldere care contin doar cifre aferente index-ului folderului.

X_train	X_test	X_validation
Y_train	Y_test	Y_validation

Setul de date este divizat in 3 subseturi:

- Antrenare: 75% din total
- Validare: 10% din subsetul de antrenare
- Testare: 25% din total

Exemple de imagini din setul de date:



3. Preprocesarea si normalizarea datelor

Fiecare imagine a fost transformata intr-un vector de octeti cu valori cuprinse intre 0 si 255. Astfel, pentru fiecare subfolder, train, validation si test, am realizat 2 vectori, X si Y, care contin octetii imaginii, respectiv eticheta fiecarei imagini.

Avem urmatorii vectori: X_train, Y_train; X_validation, Y_validation; X_test, Y_test.

Deoarece vectorii Y contin doar etichetele imaginii, am scalat doar vectorii X, impartindu-I la cea mai mare valoare, 255.

X_train = np.stack(X_train, axis=0)

X_train = X_train / 255 # scale data

Y_test = np.array(Y_test)

Astfel, setul de date a fost adus la valoarea optima.

4. Antrenarea unui model bazat pe algoritmi de invatare automata

Pentru aceasta aplicatie am utilizat algoritmul Multinomial Naïve Bayes (MultinomialNB).

In esenta, algoritmul Naive Bayes presupune ca prezenata unei anumite caracteristici intr-o clasa nu este influentata de prezenta altor caracteristici. Chiar daca aceasta ipoteza poate fi prea simplista pentru multe probleme reale, algoritmul Naïve Bayes are performante bune in multe situatii practice.

Exista mai multe tipuri de algoritmi Naïve Bayes, iar unul dintre ele este Multinomial Naïve Bayes. Acesta este utilizat in special in problemele de clasificare a textului, unde datele de

intrare sunt reprezentate sub forma de frecvente ale cuvintelor si este adecvat pentru situatiile în care caracteristicile au distributie multinomiala. De exemplu, este folosit in clasificarea textelor pentru analiza sentimentelor in recenziile de filme sau in categorizarea documentelor in domenii specifice.

In acest caz, am utilizat Multinomial Naïve Bayes pentru recunoasterea cifrelor scrise de mana. Pentru a folosi Multinomial Naïve Bayes pe setul nostru de date, se poate reprezenta fiecare imagine ca un vector de caracteristici care reprezinta pixelii unei imagini. Algoritmul estimeaza probabilitatile asociate cu valorile de intensitate a pixelilor pentru fiecare cifra.

Am antrenat algoritmul Multinomial Naïve Bayes pe 7 valori ale lui alpha: 0.001, 0.01, 0.1, 1, 10, 100, 1000, calculand pentru fiecare valoare acuratetea, pe baza subsetului de validare. Subsetul de validare are rolul de a ne ajuta sa atribuim cea mai buna valoare hyperparametrului “alpha”, cel care ne ofera cea mai mare acuratete.

Am folosit urmatoarea biblioteca din Python: MultinomialNB (sklearn.naive_bayes).

Cod pentru antrenarea algoritmului:

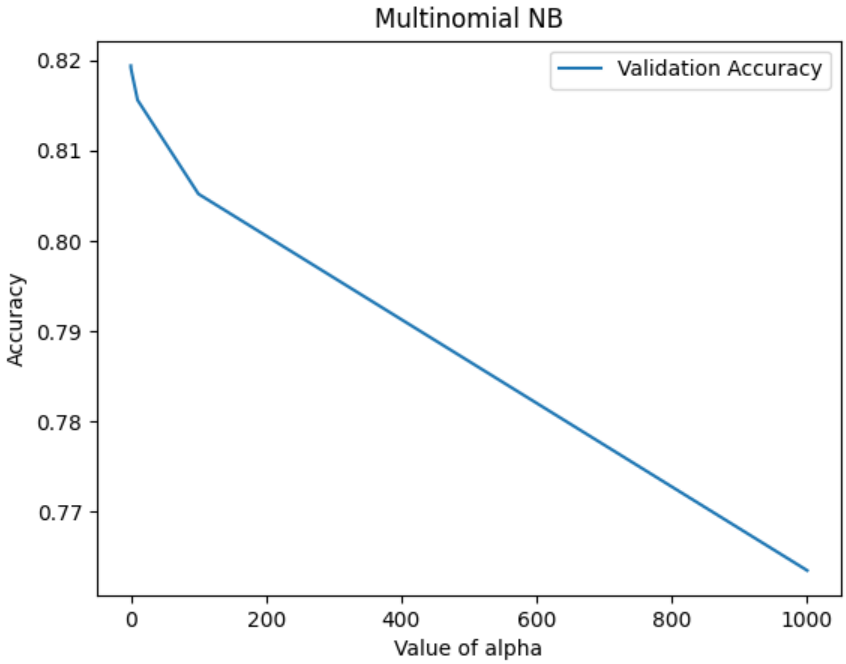
```
alphaValues = [0.001, 0.01, 0.1, 1, 10, 100, 1000]

accuracies = []

for a in alphaValues:

    # train the Multinomial Naive Bayes classifier

    nb_classifier = MultinomialNB(alpha = a)

    nb_classifier.fit(X_train, Y_train)

    # evaluate the model on validation dataset

    scoreValidation = nb_classifier.score(X_validation, Y_validation)

    # validation results

    print("alpha = %f, accuracy = %.2f%%" % (a, scoreValidation * 100))

    accuracies.append(scoreValidation)
```

5. Afisarea si interpretarea rezultatului antrenarii

Pentru fiecare valoare a lui alpha am obtinut urmatoarea acuratete:

```
alpha = 0.001000, accuracy = 81.94%
alpha = 0.010000, accuracy = 81.93%
alpha = 1.000000, accuracy = 81.88%
alpha = 10.000000, accuracy = 81.56%
alpha = 100.000000, accuracy = 80.52%
alpha = 1000.000000, accuracy = 76.35%
```

Se poate observa cum pentru alpha = 0.001 avem cea mai buna acuratete, de 81.94%. Cu cat valoarea lui alpha creste, acuratetea scade. Acest lucru a fost surprins si in urmatorul grafic:



Astfel, cel mai buna valoarea a lui alpha pentru care se poate trece la etapa de testare este 0.001.

```
alpha = 0.001000 : highest accuracy of 81.94% on validation dataset
```

Testand pentru cele 7 valori ale lui alpha, avem urmatorul timp de executie:

```
Execution time of training is:  0.739753 seconds
```

Timpul de executie aproximativ pentru fiecare valoare a lui alpha este de 0.1 secunde.

Se poate observa ca timpul de executie utilizand Naïve Bayes este mult mai mic in comparatie cu rezultatul dat de algoritmul KNN (67.7 secunde). Desi dimensiunea setului de date este mare, am obtinut un timp foarte mic de executie in etapa de antrenare. Totusi, acuratetea este mult mai mica.

Pentru algoritmul KNN: 97.24%

Pentru algoritmul Naïve Bayes: 81.94%

6. **Testarea modelului antrenat pe un set de date de test**

Testarea algoritmului s-a facut pe un set de date nou, diferit de cele de antrenare si validare.

Am ales valoarea maxAlpha = 0.001 pentru ca pentru aceasta valoare a lui alpha avem acuratetea cea mai mare.

Cod pentru testare:

```
nb_classifier = MultinomialNB(alpha = alphaValues[maxAlpha])

nb_classifier.fit(X_train, Y_train)

predictions = nb_classifier.predict(X_test)
```

7. **Afisarea si interpretarea rezultatelor testarii**

Putem observa ca am obtinut un timp scurt de executie:

```
Execution time of testing is:  0.154125 seconds
```

Urmatorul tabel de clasificare ne arata caracteristicile importante ale clasificarii, precum acuratetea, precizia, scorul F1, sensibilitatea.

Pentru fiecare cifra (0-9) avem diferite valori in aceea ce priveste precizia, sensitivitatea, specificitatea, acuratetea, astfel putem observa cat de bine a fost antrenat algoritmul Multinomial Naïve Bayes pentru fiecare cifra. Valorile pot fi influentate si de numarul de imagini din setul de antrenare/testare.

Classification Report

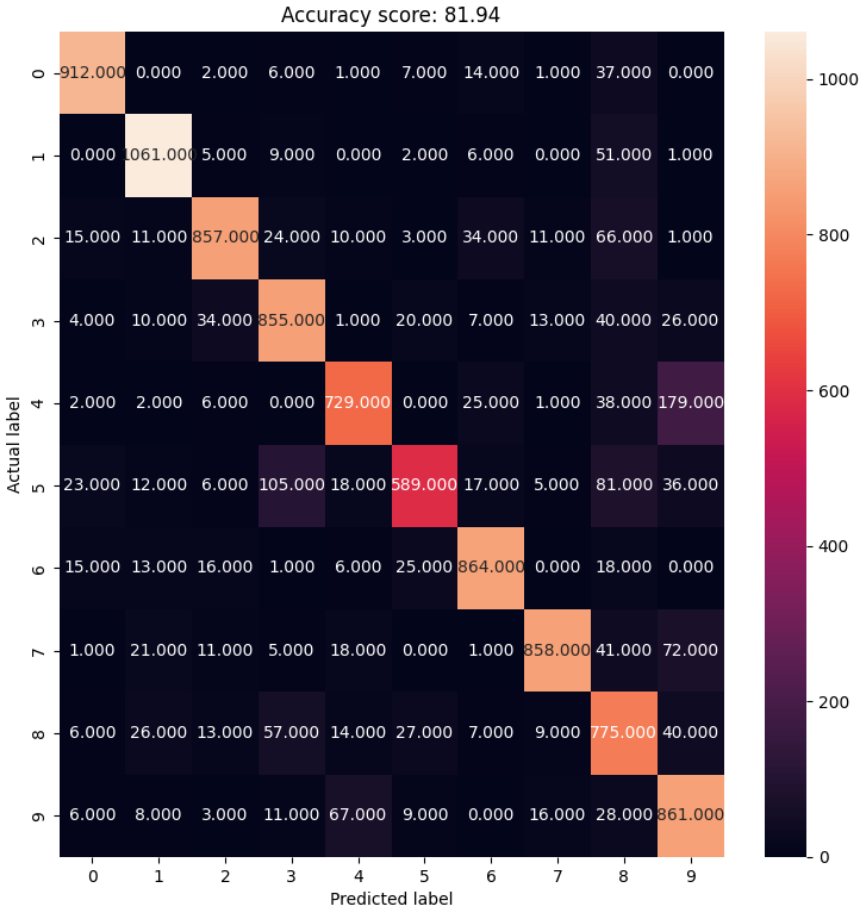
	precision	recall	f1-score	support
0	0.926829268292683	0.9306122448979591	0.9287169042769857	980.0
1	0.9115120274914089	0.9348017621145375	0.9230100043497173	1135.0
2	0.8992654774396642	0.8304263565891473	0.8634760705289671	1032.0
3	0.7968313140726934	0.8465346534653465	0.8209313490158425	1010.0
4	0.84375	0.7423625254582484	0.7898158179848321	982.0
5	0.8636363636363636	0.6603139013452914	0.7484116899618806	892.0
6	0.8861538461538462	0.9018789144050104	0.8939472322814278	958.0
7	0.9387308533916849	0.8346303501945526	0.8836251287332646	1028.0
8	0.6595744680851063	0.795687885010267	0.7212657049790601	974.0
9	0.7080592105263158	0.8533201189296333	0.7739325842696628	1009.0
accuracy	0.8361	0.8361	0.8361	0.8361
macro avg	0.8434342829089767	0.8330568712409994	0.8347132486381641	10000.0
weighted avg	0.8445434547662352	0.8361	0.8369368104168472	10000.0

Putem observa ca precizia difera in functie de cifra pe care o avem. Cea mai mare precizie: 93.87% pentru cifra 7, iar cea mai mica: 79.68% pentru cifra 3.

Matricea de confuzie este o metoda utila pentru a evalua performanta unui model de clasificare. Aceasta matrice ofera o imagine detaliata a rezultatelor predictiilor facute de model comparativ cu valorile reale din setul de date de testare.

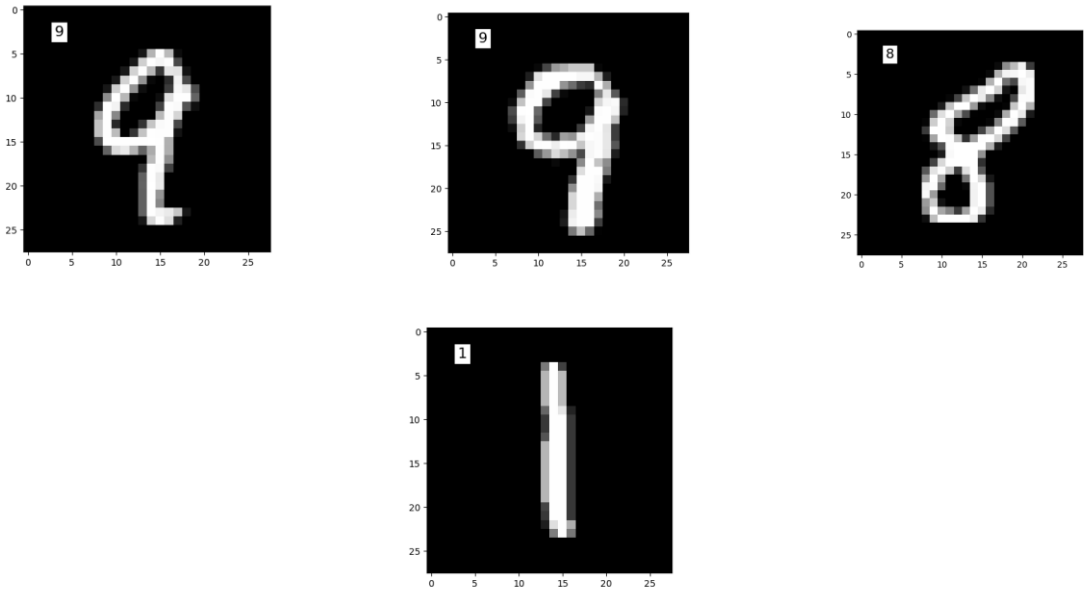
O matrice de confuzie ideala ar contine valori doar pe diagonala principala, restul fiind 0, asta ar inseamna ca nu au existat confuzii intre rezultatele din etapa testarii.

Cel mai simplu mod de a masura performanata este de a analiza in cate cazuri clasificatorul indica raspunsul corect.

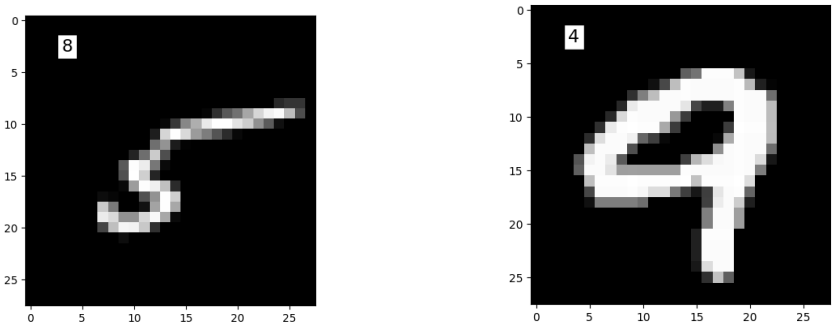


Prin intermediul matricei de confuzie putem vedea ce cifre au fost prezise in mod corect de cele mai multe ori, in functie de valoarea de pe diagonala. Astfel, cifra 1 a fost prezisa corect de cele mai multe ori, iar cifra 5 de cele mai putine ori. Totodata, se poate observa ca cifra 4 a fost confundata cu cifra 9 in multe cazuri.

Pentru a testa corectitudinea algoritmului, am comparat valoarea prezisa cu imaginea originala din setul de testare. Astfel, avand imaginea, am putut observa ce cifra a prezis algoritmul ca este:



Putem observa ca exista si cazuri in care avem erori, deci cifra prezisa este diferita fata de cea din imagine:



Putem observa cazul in care cifra 9 a fost confundata cu cifra 4, iar cifra 5 a fost confundata cu cifra 8, lucru surprins chiar si in matricea de confuzie, unde se putea observa culoarea mai proeminenta.

8. Concluzii

Algorimul Naïve Bayes ofera acuratetea de 81.94%, in comparatie cu KNN, unde am avut 97.24%. Se poate observa ca algoritmul KNN ofera o acuratete mult mai mare, in acest caz.

In schimb, timpul de executie difera, caci la Naïve Bayes am obtinut un timp de cateva secunde, 0.7 pentru antrenare si 0.13 pentru testare. Pentru KNN avem un timp de executie de 10 ori mai mare, de 67, respectiv 5 secunde pentru antrenare si testare.

Performantele celor doi algoritmi variaza in functie de setul de date si de parametri alesi. La KNN, am avut hyperparametrul k, care reprezenta numarul de vecini. Am testat pentru 10 valori ale lui k, intre 1 si 10. Pentru Naïve Bayes, am testat doar 7 valori ale hyperparametrului alpha, intre 0.001 si 1000.

Pentru acest set de date cu cifre, se pare ca algoritmul KNN a functionat mai bine, avand o acuratete mai mare. Desi, in general, algoritmul Naïve Bayes, este utilizat pentru set de date ce contine text, prin faptul ca am transformat imaginea intr-un vector de octeti, am putut aplica algoritmul pe acest set de date, obtinand un rezultat bun, doar cu o acuratete mai mica.

Am surprins cateva diferente intre cei doi algoritmi in urmatorul tabel:

KNN	Naïve Bayes
Acuratete mai mare	Acuratete mai mica
Lent	Rapid
Hyperparametrul “k”	Hyperparametrul “alpha”
Bazat pe instante	Probabilistic
Dependenta caracteristicilor	Independenta caracteristicilor
Consuma multa memorie	Eficient in stocarea datelor

Biblioteci utilizate:

```
import matplotlib.pyplot as plt

import numpy as np

from PIL import Image

from numpy import asarray

import os

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from datetime import datetime

import seaborn as sns

import pandas as pd

import cv2
```

```
from sklearn.naive_bayes import MultinomialNB
```